Instant Pascal [R] vs. ANS Pascal

This document describes the relationship between Instant Pascal and the requirements of ANSI/IEEE770X3.97-1983, American National Standard Pascal (ANS Pascal).

1: Exceptions to ANS Pascal Requirements

Instant Pascal complies with the requirements of ANSI/IEEE770X3.97-1983 with the following exceptions:

- The special symbol @ is not supported.
- In ANS Pascal, identifiers may be of any length and all characters are significant. In Instant Pascal, all characters in identifiers are significant, but the largest identifier is restricted to 255 characters.
- In ANS Pascal, a character string of length 1 is a char-type value and a character string of length *n* is a value of a packed-string-type (a ***packed array** [1..n] **of** char* -- referred to as a *string-type* in ANS Pascal) with *n* components. In Instant Pascal, all quoted character strings are string-type values. However, the compatability and assignment-compatability rules in Instant Pascal make its behavior with respect to character strings compatible with ANS Pascal.
- In ANS Pascal, no statements that *threaten* the value of a control variable of a for-statement are allowed. In Instant Pascal, this restriction is not as severe: no statement may *alter* the value of the control variable of a for-statement while the for-statement is executing.
- In Instant Pascal, only the standard file variables *input* and *output* are allowed as program parameters.
- In Instant Pascal the types ***packed record***, ***packed set***, and ***packed file*** are not supported.
- In Instant Pascal, the long forms of the required procedures ***NEW*** and ***DISPOSE*** are supported, but not implemented. Enough memory is allocated to accommodate the largest possible variant, and no assumptions are made as to which variant is currently active.
- In ANS Pascal, if no tag-field occurs in a record declaration, then an access to a field in a variant-part of that record activiates that variant-part, and all other variant-parts become inactive. In Instant Pascal, the other variant-parts do not become inactive.
- In Instant Pascal, procedural and functional parameters are not supported.
- In ANS Pascal, a component of a packed-type may not be referenced as an actual variable parameter. In Instant Pascal, this restriction is not enforced.

2: Extensions to ANS Pascal

The following Instant Pascal features are extensions to Pascal as specified by ANSI/IEEE770X3.97-1983:

- The following are word-symbols in Instant Pascal:

  otherwise string uses

- An identifier may have an underscore appearing anywhere following the initial letter of the identifier.

- Instant Pascal supports the additional integer type *longint* and the additional real-types *double, extended* and *computational*. All values of these types may be represented with numbers.

- A signed constant identifier may denote a value of type *integer*, *longint*, *real*, *double*, *extended*, or *computational*.

- In Instant Pascal, the rules for mixed *integer* and *longint* arithmetic are simple: all operands are converted to *longint* before any integer arithmetic is performed, and the result is always *longint*. A *longint* value may be used wherever an *integer* value is required if the value falls within the range *-maxint..maxint*.

- Similarly, all integer-type and real-type operands are converted to *extended* before any real arithmetic is performed and the result is always *extended*. An *extended* value may be used wherever a *real*, *double*, or *computational* value is required, provided the value falls within the range of values permissible.

- Pointer-type values may be compared for equality and inequality.

- Instant Pascal supports string-types, which are compatible with other string-types, packed-string-types, and char-type.

- In Instant Pascal, the assignment-compatibility rules have been extended to allow the mixing of string-types, packed-string-types, and char-types where appropriate.

- Individual char-type components of string-type variables may be referenced as though the string were a one-dimensional array.

- String-types may be compared with char-type and packed-string-type values.

- Instant Pascal supports a string-type value with no string elements (the null string ' ').

- Instant Pascal supports the use of the *indefinite-string-type*, i.e. the word-symbol ***string***, as a value or variable-parameter type.

- Instant Pascal includes a set of string procedures and functions.

- Instant Pascal has an optional ***otherwise*** clause for the case-statement.

- Instant Pascal supports the predefinded *library* "SANE" that may be accessed via a uses-clause.

- In Instant Pascal, functions are permitted to return non-simple types.

- An optional second parameter may be given to *reset* or *rewrite* to associate a file variable with an external file.

- Instead of *reset* or *rewrite*, a file may be opened with *open* to allow random read/write access to a file.
- An explicit *close* procedure is supplied for those file variables associated with external files.
- A *seek* procedure may be used for random access to file components.
- A *filepos* function returns the component number of the current file position, a value that may be used in subsequent *seeks*.
- String-type and enumerated-type values may be written to textfiles with *write* and *writeln*.
- "Lazy I/O" is used to permit interactive and non-interactive I/O to be treated identically.

NOTE

There is no automatic means, in Instant Pascal, of distinguishing between a program that uses extensions and one that does not.

3: Implementation-Dependent Features

The effect of using an implementation-dependent feature of Pascal, as defined by ANSI/IEEE770X3.97-1983, is unspecified. Programs using such features should not depend on any specific course being chosen, as the results may be unpredictable. This leaves the implementation free to choose the course that is most convenient at the time.

4: Treatment of Errors

This section defines those errors listed in Appendix D of the ANS Pascal standard that are NOT automatically detected and reported by Instant Pascal . The number of each error listed below is the number under which the error is listed in the standard's appendix. The wording of the description of the error, however, differs from the wording in the standard.

2. If *t* is the tag-field of a variant part, and if *f* is a field within the currently active variant part, then it is an error to attempt to alter the value of *t* while a reference to *f* exists.

4. If *p* is a pointer-type value, it is an error to reference *p^* if the value of *p* is undefined.

5. If *p* is a pointer-type value, it is an error to attempt to *dispose* of *p* while a reference to *p^* exists.

6. If *f* is a file-type variable, it is an error to close *f* or alter the current file position of *f* while a reference to *f^* exists.

19. If a pointer-type variable *p* is assigned a value by *new(p, c_1, c_2, ...c_n)*, it is an error to attempt to make any other variants than those selected by the case constants in *new* become the active variant.

20. If a pointer-type variable *p* is assigned by *new(p, c_1, c_2, ...c_n)*, it is an error to attempt to *dispose* of *p* without supplying the same list of case-constants in the same order.

21. Same as 20.

22. Same as 20.

25. If a pointer-type variable *p* is assigned a value by *new(p, c_1, c_2, ...c_n)*, it is an error to reference the entire variable *p^* in an expression, as an actual variable parameter, or as the destination of an assignment statement.

27. For *pack(a,i,z)*, it is an error if any of the referenced components of *a* have undefined values.

30. For *unpack(a,i,z)*, it is an error if any of the components of *z* have undefined values.

43. It is an error to reference a variable in an expression if the value of that error is undefined.